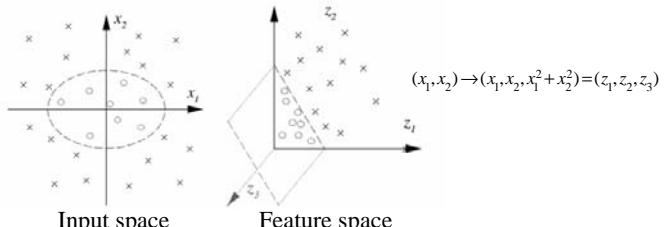


Kernel methods



- Non-linear high dimensional mapping of the data so it becomes linearly separable in the feature space
- This is equivalent to finding a non-linear decision boundary in the input space
- Computation in the feature space can be costly because it is high dimensional
 - The feature space is typically infinite-dimensional!



Kernel methods

- Suppose $\phi(\cdot)$ is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$
- An inner product in the feature space is

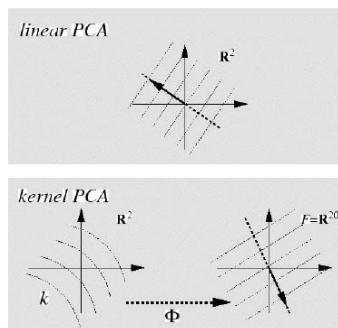
$$\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \rangle = (1 + x_1y_1 + x_2y_2)^2$$
- So, if we define the kernel function as follows, there is no need to carry out $\phi(\cdot)$ explicitly

$$K(x, y) = (1 + x_1y_1 + x_2y_2)^2$$

- This use of kernel function to avoid carrying out $\phi(\cdot)$ explicitly is known as the kernel trick. In any linear algorithm that can be expressed by inner products can be made nonlinear by going to the feature space



Kernel PCA



Kernel PCA

(Scholkopf et al., 1998)

- Eigenvectors of the cov. Matrix in feature space.
- $$\bar{\mathbf{C}} = \frac{1}{n} \sum_{i=1}^n \Phi(\mathbf{d}_i) \Phi(\mathbf{d}_i)^T \quad \bar{\mathbf{C}} \mathbf{b}_1 = \mathbf{b}_1 \lambda$$
- Using the ‘kernel-trick’ (Eigenvectors lie in the span of data in feature space $\mathbf{b}_1 = \sum_{i=1}^n \alpha_i \Phi(\mathbf{d}_i)$.)

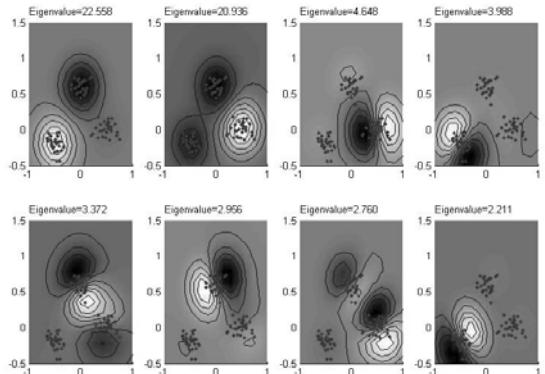
$$\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \Phi(\mathbf{d}_i) K(\mathbf{d}_i, \mathbf{d}_j) = [\sum_{i=1}^n \alpha_i \Phi(\mathbf{d}_i)] \lambda$$

$$\mathbf{K}\mathbf{a} = \mathbf{a}\lambda$$



Examples

$$\langle \mathbf{b}, \Phi(\mathbf{d}) \rangle = \sum_{i=1}^n \alpha_i K(\mathbf{d}_i, \mathbf{d})$$



Kernel Canonical Correlation Analysis

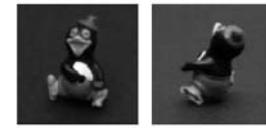


Fig. 1. Extreme views of training set

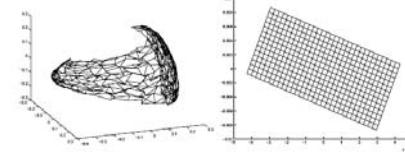


Fig. 2. Parametric manifold obtained with PCA (a) and CCA (b).

Kernel-CCA

$$\mathbf{K} = \mathbf{X}^T \mathbf{X}$$

$\mathbf{L} = \mathbf{Y}^T \mathbf{Y}$ — n × n Inner Product (Gram) Matrix

$$r = \frac{\begin{pmatrix} \mathbf{f}^T & \mathbf{g}^T \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{KL} \\ \mathbf{LK} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}}{\begin{pmatrix} \mathbf{f}^T & \mathbf{g}^T \end{pmatrix} \begin{pmatrix} \mathbf{KK} & \mathbf{0} \\ \mathbf{0} & \mathbf{LL} \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}}$$

Kernel-CCA

- Apply non-linear transformations $\phi()$, $\theta()$ to the data

$$\phi(\mathbf{X}) = \langle \phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_n) \rangle \quad \theta(\mathbf{Y}) = \langle \theta(\mathbf{y}_1), \dots, \theta(\mathbf{y}_n) \rangle$$

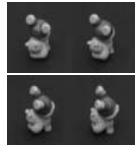
- The Kernel Trick:

$$\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = k_\phi(\mathbf{x}_i, \mathbf{x}_j) \quad \mathbf{L}_{ij} = \theta(\mathbf{y}_i)^T \theta(\mathbf{y}_j) = k_\theta(\mathbf{y}_i, \mathbf{y}_j)$$

Inner Product in Feature Space Kernel Evaluation in Input Space

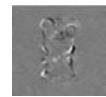
Experiments

Hippo:
Rotated through
360° (1 DOF) in
2° steps.

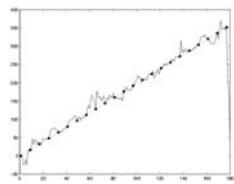


Linear Y-Encoding: $y_i = \text{turntable position } \alpha_i$ in degrees.

CCA-Factor



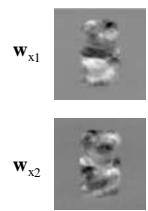
Estimates for y_i



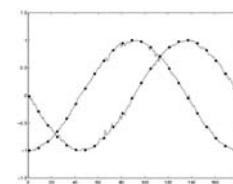
Experiments

Trigonometric Y-Encoding:
 $y_i = \langle \sin(\alpha_i), \cos(\alpha_i) \rangle$

CCA-Factors



Estimates for y_{i1}, y_{i2}



atan2

